Students' Conceptions of Programming in the Context of Game Design

Fatma Batur, Torsten Brinda

▶ **To cite this version:**

Fatma Batur, Torsten Brinda. Students' Conceptions of Programming in the Context of Game Design. Open Conference on Computers in Education (OCCE), Aug 2021, Tampere, Finland. pp.79-90, 10.1007/978-3-030-97986-7_7 . hal-04161454

HAL Id: hal-04161454
https://inria.hal.science/hal-04161454v1

Submitted on 1 Aug 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

This document is the original author manuscript of a paper submitted to an IFIP conference proceedings or other IFIP publication by Springer Nature. As such, there may be some differences in the official published version of the paper. Such differences, if any, are usually due to reformatting during preparation for publication or minor corrections made by the author(s) during final proofreading of the publication manuscript.

Students' Conceptions of Programming in the Context of Game Design

Fatma Batur and Torsten Brinda

Computing Education Research Group, University of Duisburg-Essen,
45127 Essen, Germany
{fatma.batur,torsten.brinda}@uni-due.de

Abstract. Several attempts have been made to understand novice programmers' difficulties and misconceptions in introductory programming. Most studies in the field of students' conceptions of (object-oriented) programming have only focused on identifying (mis-)conceptions without including popular contexts like game design. Since digital games are an important part of students' everyday life, exploring students' conceptions of digital games and their programming may give some recognisable patterns which might be helpful for teaching. This paper presents a brief overview of an empirical qualitative pilot study with the aim to investigate undergraduate students' conceptions of (object-oriented) programming in the context of the game Tetris®. For this study, we interviewed four students who were 19 to 21 years old, and analysed the transcripts using qualitative text analysis. Moreover, an online survey provided qualitative data from 25 participants. The first findings show that students' conceptions are based on the rules of the game, and first indications about influence factors could be found. As a result of these investigations, implications were made for the future main study.

Keywords: Conceptions, computer science, game design, digital games, programming, OOP, qualitative text analysis

1 Introduction

Programming, especially object-oriented programming (OOP), is challenging for many students. Besides, many teenagers play digital games in their free time. Therefore, designing digital games in a computer science (CS) class could be a useful context in introductory programming. Unsurprisingly, designing and creating games is a common context to teach OOP or programming in general. By exploring students' conceptions of digital games and their programming with a long-term study, we want to get some insights into the development of conceptions in the introductory class and about possible influencing factors such as teaching materials, teacher instructions, or integrated development environments (IDEs). It has previously been observed that students especially develop misconceptions of object-oriented concepts [1]. CS teachers should be able to:

identify and address students (mis-)conceptions for creating adequate learning arrangements. What is known about the investigation of students' perspectives is largely based on studies conducted in natural sciences [2]. Diethelm et al. [3] adapted the model of "Educational Reconstruction" for computer science education which emphasises that students' conceptions should be considered for course design and arrangement.

2 Theoretical Background and Related Work

2.1 Educational Reconstruction

Figure 1 provides the summary for the model of *Educational Reconstruction* adapted to computer science education (CSE) [3]. Nowadays, children are surrounded by digital technologies (which can be defined as CS phenomena), "and at an early age, children start to form conceptions of how these technologies work and their basic capabilities" [4]. Therefore, students' perspectives have to be taken into account for designing courses equally to statements by the scientific community. Particularly, this model can be used for research in order to support CS teaching by exploring students' conceptions of various topics of CS. It is important to clarify that this model focuses on students' (pre-)conceptions rather than "mis"-conceptions, as this term already conveys that ideas might be wrong or invaluable.

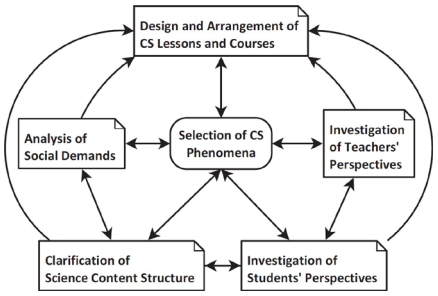

Fig. 1. Educational Reconstruction for Computer Science Education [3]

2.2 Students' Conceptions of Programming and OOP

Previous studies focused on different aspects of OOP concepts when investigating students' conceptions and mental models. For instance, Eckerdal et al. [5] describe a study of investigating students' conceptions of the OOP concepts *object and class* with a phenomenographic research approach. Other specific OOP topics were for example *objects* [6], *this operator* [7, 8], *storing objects* [9],

program state visualisations [10], *primitive and object variables* [11]. In a literature review, Qian and Lehman [1] summarised "that students exhibit various misconceptions and other difficulties in syntactic knowledge, conceptual knowledge, and strategic knowledge." [1]. They have identified several impact factors on students' conceptions: "unfamiliarity of syntax, natural language, math knowledge, inaccurate mental models, lack of strategies, programming environments, and teachers' knowledge and instruction." [1]. Besides, Xinogalos, [12] lists numerous studies on novice programmers investigated, partially contradictory, conceptions of OOP [12]. In his study he found similar conceptions as in the study of Eckerdal et al. [5]. Most of the students had the conception of "'objects' as models/entities of some real-world phenomenon and 'classes' as models describing kinds of objects." [12]. A longitudinal study of the comprehension of OOP concepts by novices by Ragonis et al. [13] report that their results can be "divided into four primary categories: class vs. object, instantiation and constructors, simple vs. composed classes, and program flow" [13].

A much-debated question is how to survey students' conceptions properly [2]. Many studies used qualitative data as interviews and/or questionnaires with open-ended questions. Concept maps [14] and diagnostic tools [7] were also used to identify misconceptions. A literature analysis revealed few studies which surveyed high school students [8, 13, 15]. Most studies in the field have only focused on novice programmers in introductory courses at colleges or universities.

Up to now, no research has been found that surveyed students' conceptions of (object-oriented) programming in the context of digital games.

2.3 Game-Design in Computer Science Education

> *Thinking about creating games means thinking about objects and how they react to one another and to the player's input. So the game creator naturally thinks in an object-oriented way.* [16]

It has been demonstrated that various reasons such as motivation, interest or reference to students' everyday life are given for using a game-based approach in teaching. For instance, Rais et al. [17] say that a game-based approach supports students' understanding of object-oriented concepts. Even games (e.g. in [18]) with the content of object-oriented concepts were developed for an introductory CS course. Xinogalos et al. [19] emphasise the importance of engaging students in introductory programming by connecting the contexts to "their interest, such as games and mobile apps" [19]. In "Does Computer Game Design and Programming Benefit Children?" Denner et al. [20] analysed over 400 articles with the result that "computer game design and programming can lead to changes in programming knowledge, problem solving, and computer science attitudes and confidence." [20]. In another study conducted by Troiano et al. [21], it was shown that the game genre supports students' development of computational thinking skills. Besides, it helps students to focus on game design. Moreover, the researchers indicate to consider the impact of game genre while teaching

programming with a game-based approach. To conclude this section, the literature identifies the lack of research in students' conceptions of designing and programming games.

2.4 Research Questions

The model of Educational Reconstruction suggests that successful CS courses should be designed by considering students' perspectives, and that game-based approaches in CSE are relevant for teaching introductory programming. Therefore, the research questions in this study focus on programming (OOP) conceptions of novice programmers in the context of digital games. Interviews will be conducted at several different points during a programming introductory course. The long-term investigation of students' developed conceptions may will also give insight into possible influencing factors. In particular, this study will examine the following main research question:
Which conceptions do students develop in different phases of introductory programming when learning OOP in the context of game design? The research question is divided into:

- Which conceptions do learners have about digital games and their functions?
- Which programming-related conceptions do learners develop about digital games and their functions and about (relevant) object-oriented concepts?
- How do programming environments/teacher instructions/materials influence students' conceptions of OOP?
- How does previous knowledge of programming/algorithms influence students' conceptions of OOP?

3 Study design

The use of qualitative case studies is a well-established approach to investigate students' conceptions [2] and is a main part of the Educational Reconstruction for CSE (see chapter 2.1). This investigation uses longitudinal data from semi-structured interviews to explore possible influence factors on students' conceptions. The semi-structured approach was chosen in order to compare the data of each individual and among each other. Furthermore, a questionnaire with open-ended questions generates qualitative data in addition to the interviews. The study uses qualitative analysis by using the method of Kuckartz [22] in order to gain insights into categorised conceptions.

3.1 Pilot Study

The pilot study was conducted in order to test the method and the developed instruments for investigation of students' perspectives. Particularly, it was important to check if the questions were chosen adequately. For the pilot study, the game Tetris® was chosen because it is well-known[1], has simple rules and a 2D game structure. Therefore, programming Tetris® could be a suitable task for undergraduate students. By surveying in two phases at the beginning and at the end of a course, the development of students' conceptions was conducted. This paper presents the results of the first surveys. The research questions were specified for the pilot study as follows:

- Which associations of the term "programming" do novice programmers have?
- Which conceptions of Tetris® do novice programmers have?
- Which conceptions of (object-oriented) programming do novice programmers have about Tetris® as an example for a digital game?

Data Collection. The data was gathered via an online questionnaire, and interviews using an online video conference tool. It was carried out with an introductory course of Media Informatics at a college in Germany, in November 2020.[2] In total, four interviews and 25 questionnaires (19 filled-in completely) were used for the analysis. 49% (14) of the students were male, 41% (12) female,and 10% (3) made no indication of gender. Almost half of the participants had no previous school experience in the field of computer science. Several limitations to this pilot study need to be acknowledged. The sample size is small, probably due to excessive demand in the first semester and COVID-19 pandemic. Only online observations of exercises were possible due to the pandemic and distance teaching. The interviews were conducted with a video conference tool, where social interactions were limited.

4 Evaluation

The structured **questionnaire** has four parts: (1) general personal questions (age, gender, prior experience in computer science education), (2) open-ended questions "What do you associate with the term 'programming'?", (3) open-ended questions "How would you program this game? Please describe a possible approach by drawing or writing your ideas." and (4) questions about gaming in everyday life, and self-evaluation in programming skills and digital media usage. The **interviews** were semi-structured with a prepared guideline[3]. For the evaluation of the four interviews, we used the qualitative text analysis of Kuckartz [22]. By applying the first two steps, the interviews have the following main logical categories:

1. Concept of the term "programming"
2. Explanation of the game Tetris®
3. Explanation of elements in the game Tetris®
4. How to program Tetris®
5. Main game situations in Tetris®
6. Add-ons for Tetris®
7. References to the course contents
8. General personal information

Within the questionnaire and the interview, the students had the option to play[4] Tetris® for several minutes.

The following evaluation shows selected results of the qualitative text analysis. All the analyses were carried out using the software MAXQDA 2020. It is important to mention that some sections were coded twice.

---

[1] "one of the best-selling video game franchises of all time" [https://en.wikipedia.org/wiki/Tetris]

[2] The course OOP 1 (based on *objects-later*) has started in October 2020, so the students have attended the course already for 3 - 4 weeks.

[3] See Appendix A.

4.1 Associations of the Term *Programming*

In figure 2, to the question "What do you associate with 'programming'?" overall 26 [4 interviewed and 22 surveyed students] responses are grouped into 82 categories. This means that the responses were very diverse. About a third of the respondents stated "to code", "programs" and/or "programming languages". Generally, the responses may be divided into following four aspects: **close to everyday life** (e.g. "games, smart home, robots, future"), **application-oriented** (e.g. "websites, apps, server, systems"), **technical view** (e.g. "loops, variables, functions, Java, C#") and **focus on planning-processes** (e.g. "pair programming, planning, testing, functional specifications, requirement specifications"). Interestingly, in only four answers we could find the term "object-oriented programming". All of them, can be classified with **technical view**. No more analyses are feasible due to incomplete details of prior experiences in computer science education.

4.2 Conceptions of Game Design

The overall response to the online questionnaire was poor. There were 18 responses to the question, "How would you program this game? Please describe a possible approach by drawing or writing your ideas." In contrast, the four interviewees responded their ideas extensively.

Table 1 illustrates the proportion of categories found. Almost 30% of those surveyed indicated that they do not have any idea. The themes of the **process** of designing Tetris and **implementing** of Tetris® recurred throughout the dataset. Some subcategories could be found in **implementing**: statements related to programming (**programming oriented**), the **bricks** and other **elements** of Tetris®. Several responses indicate conditionals, while explaining the design of the game execution. Just two participants used terms of OOP for their explanation. Furthermore, most of the statements were made from a *user's view* rather than *developer's view*. Overall, these results indicate that the majority of the students explained Tetris® by using their gaming experiences. It is therefore possible that the participants had almost no ideas of designing and/or programming a game before asking them. This result may be explained by the fact that at least 60% of those who were surveyed indicated not having any CS experience.

---

[4] https://save-society.org/browser-game/tetris/

Fig. 2. Tag cloud of found categories of the term " programming" [N=26]

4.3 Conceptions of the Game Tetris®

Table 2 lists some details of interviewees, such as age, gender and former CS experience[5]. Altogether, three identified categories *bricks*, *game execution* and *anthropomorphism of computers* are presented.

Bricks. First of all, three interviewees (A, B & D) stated that the *appearance* of the bricks should have been defined. For example, a brick is formed by smaller parts like *pixels* or *squares*. One participant commented: "How are the pixels arranged for the specific brick?" [A]. Other responses included: "One brick is built with small squares. So, I would first define a square." [D]. These two conceptions show the idea of composition and may reveal an implicit object-oriented way of thinking. A common view amongst interviewees was that the following features of a brick can be changed: its *orientation* by turning it, the *position* (motion to left or right) or the *speed* while moving top-down. As one interviewee put it: ". . . a brick moves constantly in the same speed except the user speeds it up with the arrow keys. . . " [D]. Interestingly, the same participant stated that as soon as a brick does not move any more, it has reached the bottom and no more changes of that specific brick are possible. This statement may indicate the object-oriented concepts of *object* and/or *message passing* [23]. In one case, the participant B said that the game should also know "how many bricks to use and that bricks do not overlap" [B]. This may indicate the conception of a *defined number* of bricks and the characterisation of a brick with *constraints*.

[5] In the following text the quoted person is written in square brackets after the quotation.

Table 1. Conceptions of designing the game Tetris®: categories and examples (loosely translated from German) [N=22]

Cat.		Subcategory	Example	Total
process		design	"...set the UI and the constructionby drawing up a mock-up..." [P21]	1
		planning	"...thinking about the elements and the requirements..." [P7]	9
		investigation	"...to get information about tools online..." [C]	5
		cooperation	"...by asking others for help..." [P19]	3
implementing	oriented programming	conditionals	"...delete a row, when its full..." [P13]	9
		algorithm	"calculating the score" [P13]	8
		testing	"generating automated tests" [P21]	3
		language	"...it depends on the selected programming language and its library..." [A]	2
		OOP	"...in a class diagram...required methodsand variables..." [P7]	2
		complexity	"...easy to run by the computer because no complex 3D models..." [A]	2
	brick	creating bricks	"...defining shapes for the bricks..." [P24]	9
		navigation	"...process player inputs (cursor keys) to move the bricks." [P24]	9
		random generator	"Select next object randomly..." [P8]	5
		speed	"define the speed of the bricks while moving" [P24]	3
		bricks in motion	"...the bricks should move from top to bottom..." [D]	5
	elements	grid	"some kind of a table to use as the game field" [P23]	6
		scores	"...high score should be saved..." [P7]	6
		scoreline	"...scoreline increases when getting more scores" [P1]	3

Table 2. Details of interviewees

	age	gender	CS experience
A	19	m	5 years CS at school
B	21	f	1 year CS at school and some YouTube learning videos
C	20	f	no CS at school, but tried to learn coding with an online course
D	21	f	no CS at school, no experience

Game Execution. A variety of perspectives were expressed to describe different game situations. For example, one interviewee said: ". . . every pixel needs to be diminished with a specific value as it moves down on the y-axis." [A]. This view reveals the idea of the game field corresponding with bricks. Talking about this issue, another interviewee said: ". . . control if a row is occupied with bricks, then you can disappear [delete the bricks]." [D] Furthermore, the same participant thought that the bricks move nonstop top-down "without the opportunity of stopping them because otherwise the game would never end" [D]. In a similar way, another student said: ". . . there is a 'main-loop' for execution of everything . . . or for checking the user inputs. . . " [A]. In all cases, the students reported different game procedures by using *conditionals* which indicates concepts of *imperative programming*. As one interviewee said: ". . . *if* the bricks are outside the field, *then* the game is over." [C]. Another interviewee, when asked about the end of the game, explained: ". . . the game recognises, that the border is reached, so it does not run anymore" [B].

Anthropomorphism of Computers. In all cases, the students assigned human characterisations to the computer: "the computer needs to record" [A], "the game identifies" [B], "the program should know" [C], "the program sees" [D] or "the computer needs to know" [D]. In one case, phrasing the questions with personification of the computer "Do you have an idea, which information the computer needs to move [the bricks] to the left or right?" helped to evoke the student's conceptions: "He definitely would need the number of the fields and if they are unoccupied. . . " [D].

5 Discussion

While surveying the students about programming Tetris®, we recognised that the responses have been shallow. By asking the interviewees about specific game situations, e.g. "What is happening while a brick is falling down?", conceptions of game execution could be revealed in more detail. To summarise, these results show that students' conceptions are based on the rules of the game. One interesting finding is, when asked about possible add-ons for Tetris®, all of the students thought about *special bricks* which act as a *bomb* like in *Candy Crush Saga*. Overall, these results indicate that the investigated conceptions are very specific to the game Tetris®. Therefore, these findings are rather difficult to interpret as general conceptions for digital games, even though some of them could be adapted, e.g. "main loop" [A], to other digital games. Unsurprisingly, student A with prior knowledge in programming showed more concrete conceptions. Perhaps, the students will develop more conceptions about digital games by gaining more programming experience.

6 Conclusion and Further Work

First results of the pilot study show insights into students' conceptions of game design in the example of Tetris®. Furthermore, the current study found that students possess implicit imperative programming and object-oriented concepts. However, the exploration of students' conceptions of programming by using a game-based approach showed some weak points, e.g. conceptions based on specific characteristics of Tetris®. The use of the open-ended questionnaire was not meaningful for our research goal. Therefore, we will focus on interviews in the main study.

After interviewing the students at the end of the introductory class, the evaluation will be consolidated. It is expected that object-oriented concepts will have a stronger influence. By finalising the evaluation of the pilot study, the interview guidelines will be optimised for the main study. In particular, we will add explicit questions about digital games in general, and will use Tetris® as an example and not as the focus of the interviews. In the main study, high school students will be interviewed at least three times (at the start, in the middle and at the end of the course) during the introductory class with the aim of getting more detailed conceptions related to object-oriented programming in the context of game design.

References

1. Qian, Y., Lehman, J.: Students' misconceptions and other difficulties in introductory programming. ACM Transactions on Computing Education 18(1), 1–24 (2017).
2. Duit, R., Gropengießer, H., Kattmann, U., Komorek, M., Parchmann, I.: The model of educational reconstruction – a framework for improving teaching and learning science1. In: Jorde, D., Dillon, J. (eds.) Science Education Research and Practice in Europe, pp. 13–37. Cultural Perpectives in Science Education,SensePublishers, Rotterdam, Netherlands (2012).
3. Diethelm, I., Hubwieser, P., Klaus, R.: Students, teachers and phenomenaa: Educational reconstruction for computer science education. In: Laakso, M.J., McCartney, R. (eds.) Proceedings of the 12th Koli Calling International Conference on Computing Education Research - Koli Calling '12. pp. 164–173. ACM Press, New York, NY (2012).
4. Eckerdal, A., Thuné, M., Berglund, A.: What does it take to learn 'programming thinking'? In: Anderson, R., Fincher, S.A., Guzdial, M. (eds.) Proceedings of the1st International Computing Education Research Workshop. pp. 135–142. ACM Press, New York, NY (2005).
5. Eckerdal, A., Thuné, M., Berglund, A.: What does it take to learn 'programming thinking'? In: Anderson, R., Fincher, S.A., Guzdial, M. (eds.) Proceedings of the1st International Computing Education Research Workshop. pp. 135–142. ACM Press, New York, NY (2005).
6. Holland, S., Griffiths, R., Woodman, M.: Avoiding object misconceptions. SIGCSE Bull 29(1), 131–134 (1997).
7. Ragonis, N., Ronit, S.: A diagnostic tool for assessing students' perceptions and misconceptions regards the current object "this". In: Pozdniakov, S.N., Dagienė, V. (eds.) Informatics in Schools. Fundamentals of Computer Science and Software Engineering. pp. 84–100. Theoretical Computer Science and General Issues, Springer International Publishing and Imprint, Springer, Cham, Switzerland (2018).
8. Shmallo, R., Ragonis, N.: Understanding the "this" reference in object oriented programming: Misconceptions, conceptions, and teaching recommendations. Edu-cation and Information Technologies pp. 1–30 (2020).
9. Sorva, J.: Students' understandings of storing objects. In: Proceedings of the Seventh Baltic Sea Conference on Computing Education Research - Volume 88. pp. 127–135. Koli Calling '07, Australian Computer Society, Melbourne, VIC (2007).
10. Sajaniemi, J., Kuittinen, M., Tikansalo, T.: A study of the development of students' visualizations of program state during an elementary object-oriented programmingcourse. J. Educ. Resour. Comput. **7**(4) (Jan 2008).
11. Sorva, J.: The same but different students' understandings of primitive and object

variables. In: Proceedings of the 8th International Conference on Computing Education Research. pp. 5–15. Koli ’08, ACM, New York, NY (2008).
12. Xinogalos, S.: Object-oriented design and programming: An investigation of novices’ conceptions on objects and classes. Trans. Comput. Educ. 15(3), 13:1– 13:21 (2015).
13. Ragonis, N., Ben-Ari, M.: A long-term investigation of the comprehension of oop concepts by novices. Computer Science Education 15(3), 203–221 (2005).
14. Sanders, K., Boustedt, J., Eckerdal, A., McCartney, R., Moström, J.E., Thomas, L., Zander, C.: Student understanding of object-oriented programming as expressedin concept maps. In: Dougherty, J.D., Rodger, S. (eds.) Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education. p. 332. ACM Press, New York, NY (2008).
15. Teif, M., Hazzan, O.: Partonomy and taxonomy in object-oriented thinking. In: Unknown (ed.) Working Group Reports on ITICSE on Innovation and Technology in Computer Science Education. p. 55. ACM, New York, NY (2006).
16. Overmars, M.: Teaching computer science through game design. Computer 37(4), 81–83 (2004).
17. Rais, A.E., Sulaiman, S., Syed-Mohamad, S.M.: Game-based approach and its feasibility to support the learning of object-oriented concepts and programming. In: Harun, M.F. (ed.) 5th Malaysian Conference in Software Engineering (MySEC), 2011. pp. 307–312. IEEE, Piscataway, NJ (2011).
18. Wong, Y.S., Yatim, M.H.M., Tan, W.H.: Use computer game to learn object- oriented programming in computer science courses. In: Global Engineering Education Conference (EDUCON), 2014 IEEE. pp. 9–16. IEEE (4/3/2014 - 4/5/2014).
19. Xinogalos, S., Satratzemi, M., Malliarakis, C.: Microworlds, games, animations, mobile apps, puzzle editors and more: What is important for an introductory programming environment? Education and Information Technologies 22(1), 145–176 (2017).
20. Denner, J., Campe, S., Werner, L.: Does computer game design and programming benefit children? a meta-synthesis of research. Trans. Comput. Educ. 19(3), 1–35 (2019).
21. Troiano, G.M., Chen, Q., Alba, Á.V., Robles, G., Smith, G., Cassidy, M., Tucker-Raymond, E., Puttick, G., Harteveld, C.: Exploring how game genre in student- designed games influences computational thinking development. In: Bernhaupt, R., Mueller, F.F., Verweij, D., Andres, J., McGrenere, J., Cockburn, A., Avellino, I., Goguey, A., Bjørn, P., Zhao, S., Samson, B.P., Kocielnik, R. (eds.) Proceedingsof the 2020 CHI Conference on Human Factors in Computing Systems. pp. 1–17. ACM, New York, NY (2020).
22. Kuckartz, U., Radiker, S.: Analyzing qualitative data with MAXQDA: Text, audio, and video. Springer, Cham, Switzerland (2019).
23. Armstrong, D.J.: The quarks of object-oriented development. Commun. ACM 49(2), 123–128 (2006).

A Interview guideline

The questions of part I will be asked on the different survey dates recurrently,whereas part II will be asked just the first time.

Part I - Game Design:

1. What do you associate with the term “programming”?
2. Play the game *Tetris*®.
3. Do you know this game? How would you explain it to a friend? What is the goal of the game?
4. How would you describe the different parts of the game? Which function could they have?
5. How would you program this game? Please describe a possible approach by drawing or writing your ideas.
   - Which function could have the playing field?
   - How could the controls work?
   - How does the game start or end? Can you describe it in detail?
6. Additional: Which expansion could you imagine for this game? How could you implement it/them?
7. Later on: Which topics of the class have caused the change of your conception(s)/idea(s)? Was there anything in particular that helped you construct this new understanding?
8. What features and content of the learning materials presented in the class did you find beneficial to your understandings of programming concepts? How did it (they) help your learning experience?
9. So far, how would you evaluate your programming skills? Choose a number between 0 for very low and 10 for very high.

Part II - Individual Background:

1. Do you play digital games in your free time? If yes, how often and with which devices?
2. Do you have any experience in programming? Have you ever had any CS classes or programming courses?